Dürr, C., Hanzálek, Z., Konrad, C., Seddik, Y., Sitters, R., Vásquez, O. C., & Woeginger, G. J. (2018). The triangle scheduling problem. *Journal of Scheduling*, *21*(3), 305-312. https://doi.org/10.1007/s10951-017-0533-1

The Triangle Scheduling Problem

Christoph Dürr[1], Zdeněk Hanzálek[2], Christian Konrad[3], Yasmina Seddik[2], René Sitters[4], Óscar C. Vásquez[5], and Gerhard Woeginger[6]

[1] Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6, Paris, France
christoph.durr@lip6.fr
[2] FEE and CIIRC, Czech Technical University in Prague, Czech Republic
zdenek.hanzalek@cvut.cz, yasminaseddik@gmail.com
[3] Department of Computer Science and DIMAP, University of Warwick, United Kingdom c.konrad@warwick.ac.uk
[4] Dept. of Econometrics and Operations Research, Vrije Universiteit, The Netherlands
r.a.sitters@vu.nl
[5] Dept. of Industrial Engineering, University of Santiago of Chile
oscar.vasquez@usach.cl
[6] Dept. of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands, g.woeginger@tue.nl

Abstract. This paper introduces a novel scheduling problem, where jobs occupy a triangular shape on the time line. This problem is motivated by scheduling jobs with different criticality levels. A measure is introduced, namely the *binary tree ratio*. It is shown that the greedy algorithm solves the problem to optimality when the binary tree ratio of the input instance is at most 2. We also show that the problem is unary NP-hard for instances with binary tree ratio strictly larger than 2, and provide a quasi polynomial time approximation scheme (QPTAS). The approximation ratio of Greedy on general instances is shown to be between 1.5 and 1.05.

1 Introduction

Mixed-criticality Scheduling. In a mixed-criticality system, tasks with different criticality levels coexist and need to share common resources, such as bandwidth in a communication channel [10,9] or execution time on a machine [15,1,2,8,12,13]. Contrary to single-criticality systems, the estimated worst-case executing time (WCET) of a task depends on its criticality level (the higher the criticality level, the more time is estimated). Often, however, the actual processing times of tasks are not known beforehand and the estimated WCET deviate hugely from the actual processing time. The goal is therefore to design *robust* schedules that are able to tolerate runtime variations to a reasonable extent. More conservative WCET estimates are usually used for highly critical tasks (e.g. braking in a car), while less conservative estimates suffice for low-critical tasks (e.g. displaying the engine temperature in a car). In case the allocated time for a task is insufficient at runtime, i.e., the actual runtime of a task exceeds its estimated processing time, the execution of the task may nevertheless continue

and suppress subsequent tasks of lower criticality. For a recent thorough survey on mixed-criticality systems and arising scheduling problems we refer the reader to [4].

The Triangle Scheduling Problem. In this paper, we consider the problem of non-preemptively scheduling n unit length jobs/tasks with different criticality levels on a single machine. The objective is to minimize the *makespan* (maximum completion time). Let $p_i \in \mathbb{N}$ denote the criticality level of job i. The expected execution time of every job is 1. If however at runtime, a job i requires more time — which can be any fraction greater than 1 — then we continue its execution for at most p_i time units, and other jobs with lower criticality levels that were scheduled in these slots are canceled. The computed schedule has to fulfill the following properties:

1. [*Priority*] When prolonging the execution of a job, no other job with equal or larger criticality level needs to be canceled.
2. [*Fairness*] If a job i with criticality level p_i is canceled due to prolongation of job j, then j is executed for at least p_i time steps.

The priority property ensures that a job never affects the execution of a job of equal or higher criticality, thus signifying that highly critical jobs are prioritized. The fairness property ensures that if a job i is canceled due to the prolonged execution of job j, then the prolongation of j is substantially large (from one time unit to at least p_i time units). This property ensures that highly critical jobs are canceled only in rather exceptional circumstances: It is not only necessary that the execution of an even more critical job is longer than the initially estimated one time unit, but the actual execution time is at least p_i times as long as initially assumed.

To illustrate the previous properties, consider a four jobs instance with criticality levels $p_1 = 6$, $p_2 = 5$, $p_3 = 4$ and $p_4 = 3$. The schedule that places job i into time slot $i - 1$ — see Figure 1 —respects the priority property (the prolongation of any job wouldn't cancel other jobs of higher criticality), however, it violates the fairness property: If, for example, job 1 is prolonged by only a single time slot, then job 2 needs to be canceled despite its rather high criticality level of 5, which violates the fairness property. The optimal schedule for this instance schedules job 1 in time slot 0, job 3 in slot 4, job 4 in slot 7, and job 2 in slot 10.

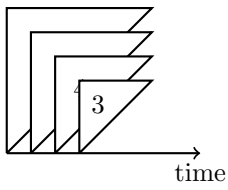

Fig. 1. Example of an infeasible schedule.

To obtain a handle on the previous scheduling problem, we identify that it can be see as a one-dimensional triangle alignment problem, defined as follows:

Definition 1 (Triangle scheduling, Gap). *Given integers $p_1, \ldots, p_n$ with $p_1 \geq \ldots \geq p_n > 0$ find starting times $s_1, \ldots, s_n \geqslant 0$ minimizing the so-called* makespan $\max_j s_j + p_j$ *such that for all $i \neq j$ we have $|s_i - s_j| \geqslant \min\{p_i, p_j\}$. We call* gap *any interval spanned by successive starting times, including the interval between the maximum starting time and the makespan of the schedule.*

We abbreviate the triangle scheduling problem as TS. Figure 2 illustrates the previously mentioned four jobs instance.

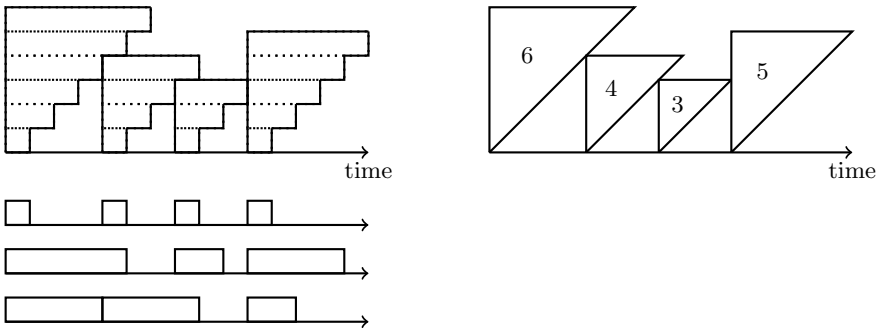

Fig. 2. Example of an optimal schedule. **Left:** discrete version. **Below:** different possible executions. In the first one each job requires a single time slot and all jobs are executed. In the second and third executions, jobs require longer processing times preventing the execution of some jobs with lower criticality. **Right:** continuous version. Each triangle corresponds to a job j labeled with its criticality level p_j. The left border of a triangle is the starting time of the job, while the right end is its worst-case completion time.

Our Contributions. In this paper, we initiate the study of TS. We show that the problem is strongly NP-hard, which implies that TS does not admit a fully polynomial time approximation scheme (FPTAS), unless $P = NP$. We provide a quasi polynomial time approximation scheme (QPTAS), which implies that TS is not APX-hard unless P=NP. In addition we present a Greedy algorithm, which processes the triangles from largest to smallest, placing each into a currently largest gap and potentially shifting subsequent triangles to the right if the gap is not large enough to contain it. We show that this algorithm has an approximation factor between 1.05 and 1.5.

Binary Tree Ratio. Furthermore, we establish a measure, that allows us to distinguish hard from easy instances:

Definition 2 (Binary tree ratio). *Given an instance of* TS $p = p_1, \ldots, p_n$ *with* $p_1 \geq \ldots \geq p_n > 0$*, we define its* binary tree ratio $R(p)$ *as*

$$R(p) := \max_{i=2,\ldots,n} p_{\lceil i/2 \rceil}/p_i.$$

Schedules computed by our Greedy algorithm can be represented by binary trees on the jobs of the problem instances (see Section 2.3 details). The binary tree ratio is the maximum ratio between a job and its successor in the tree. We will show that our Greedy algorithm solves an instance to optimality if its binary tree ratio is at most 2. On the other hand, we prove that there are instances with binary tree ratio strictly larger but arbitrarily close to 2 that render the problem

NP-hard. A binary tree ratio of 2 is hence the cut-off point that separates hard from easy instances.

Other Related Works. Since a few decades the scheduling community has been very interested in producing robust schedules that can react to changes in job characteristics. For example in [11] a model is studied where the processing times can vary, and a schedule has to be produced with good objective value even under these variations. In [1] an algorithm is given for a slightly different model (preemptive schedule, release times, arbitrary processing times per critical level) which constitutes a 1.618-approximation algorithm for the special case of 2 critical levels. In contrast the problem studied in this paper however has a trivial optimal solution in case $p_j \in \{1, 2\}$ for all jobs j. For more information see the survey [3] as well as a recent PhD thesis and references therein [16].

Vestal [15] introduced the *mixed-criticality framework*, where the execution of lower critical jobs can be canceled in order to grant high criticality jobs the necessary amount of resources. Applications are mostly embedded systems. In a communication system, jobs represent messages, and safety-critical messages have to co-exist with less critical ones that are not subject to hard constraints. For instance, the IEC 61508 standard defines four Safety Integrity Levels (SIL) (e.g. the importance of a safety-related job performing braking in a car is much higher than the importance of a job displaying the engine's temperature). Similarly the *CANaerospace* protocol specifies several criticality levels for messages, and in order to guarantee delivery times of high critical messages, the transmission of lower critical messages can be canceled [14]. The static schedule with multiple variants of has been studied also for the *FlexRay* protocol used in modern cars [6].

Our model can be seen as a special case of the message transmission model described by Hanzálek et al. in [10]. They consider a single machine scheduling problem with release times, deadlines, different criticality levels, and WCETs that depend on the task and the criticality level. They propose a linear programming formulation of the problem and prove NP-completeness of this more general problem. Note that our model does not consider release times and deadlines, and the WCET of a task equals its criticality level.

Finally we would like to mention a connection with the computational problem of packing triangles in a given rectangle, which has applications in industrial cutting and storage. The later has been shown to be NP-hard [5], while our paper shows that the problem is already hard in the particular case of right triangles that can only be vertically translated and not rotated.

Outline. In Section 2 we consider our Greedy algorithms. Then, in Section 3, we show that solving the problem for instances with binary tree ratio strictly larger than 2 is strongly NP-hard. Last, we provide a quasi polynomial time approximation scheme (QPTAS) in Section 4.

2 The Greedy algorithm

We propose a polynomial time approximation algorithm denoted *Greedy*. Recall that jobs are sorted with respect to their criticality levels, i.e., $p_1 \geq p_2 \geq \cdots \geq p_n$.

Definition 3 (Greedy). *Job 1 starts at time $s_1 = 0$. This creates a unique gap from time 0 to time p_1, see Definition 1. Then, every job $j = 2, \ldots, n$, is placed in a largest gap (the first one in case of tie). If the chosen gap has length x and starts at time s_i, then the current job j is placed at $s_j = s_i + p_j$. If $2p_j > x$, then all jobs k with $s_k > s_j$ are delayed by $2p_j - x$ in order to maintain feasibility, see Figure 3.*

Note that in case $2p_j > x$ the makespan increases by $2p_j - x$. Hence by choosing the largest gap, Greedy minimizes the increase of the makespan at every step.

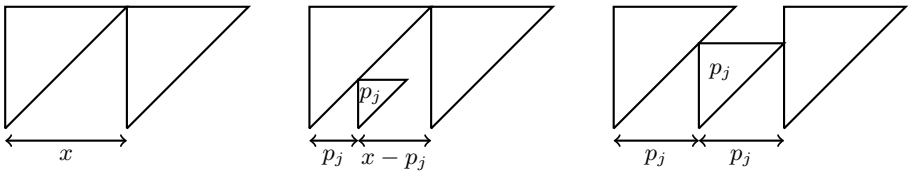

Fig. 3. When Greedy inserts a job j in a gap of size x (left figure) it creates two gaps. One of size p_j and another either of size $x - p_j$ if $x \geq 2p_j$ (center figure) or of size p_j if $x < 2p_j$ (right figure).

2.1 Lower bound on the optimum

A simple lower bound can be obtained by relating gaps to jobs in the schedule and using the fact that a gap between jobs i and j has size at least $\min\{p_i, p_j\}$.

Lemma 1. *Let $S = p_{\lceil n/2 \rceil + 1} + \ldots + p_n$ be the total processing time of the smaller half of the jobs, and $m = p_{(n+1)/2}$ if n is odd and $m = 0$ if n is even. The optimal makespan OPT is at least*

$$m + 2S.$$

Proof. Consider an arbitrary feasible schedule, and let T be its makespan. To obtain a bound on T, we charge every gap to a job as follows:

Map every gap between two consecutive jobs i, j to the smaller job among them, breaking ties arbitrarily. Map the last gap between job j and the makespan to job j. Now every job j is the image of 0, 1 or 2 gaps in this mapping. Let $a_j \in \{0, 1, 2\}$ be this number. There are exactly n gaps, hence we have

$\sum_{j=1}^{n} a_j = n$. Moreover, since every gap is mapped to a job of no larger size and the total gap size is T, we have

$$\sum_{j=1}^{n} a_j p_j \leq T.$$

The proof follows from the fact that the left hand side is minimized when $a_j = 0$ for the larger half of the jobs, $a_j = 2$ for the smaller half of the jobs and possibly $a_{(n+1)/2} = 1$ if n is odd. □

Note that this lower bound can be very weak. Consider a 2-job instance with $p_1 = M$ and $p_2 = 1$, for a large $M \geq 2$. Then the lower bound states that the makespan is at least 2, while the optimal makespan is M.

2.2 Approximation ratio of Greedy

Lemma 2. *The approximation ratio of Greedy is at most* 1.5.

Proof. Consider the final schedule produced by Greedy on instance p. We perform the following transformations:

First, we delay each job by as much as possible, while maintaining feasibility, the makespan and the job order. This transformation might change the order of the gap sizes, but does not modify the actual gap sizes when viewed as a multi-set.

Second, we define a *truncated* instance p' as follows. For every job j which starts at some t followed by a gap of size x with $x < p_j$, we set $p'_j = x$. For all other jobs j, we set $p'_j = p_j$, see Figure 4. Makespan as well as feasibility of the schedule are preserved by the truncation.

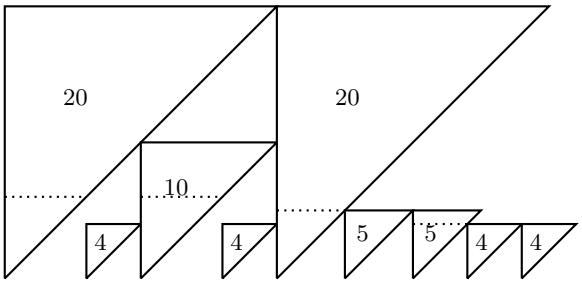

Fig. 4. Transformations on a schedule produced by Greedy: Jobs are delayed (right-shifted) and their sizes truncated (dotted lines).

We assume that Greedy increased the makespan when placing the last job n. This assumption is without loss of generality, since removing jobs that followed

the last makespan increase only decreases the makespan of the optimal schedule, while the makespan produced by the algorithm is preserved.

We claim that for all i we have $p_n \leq p'_i < 2p_n$. Indeed when job n was placed, all gaps were of size strictly less than $2p_n$ since the insertion of job n increased the makespan. Furthermore, by induction, it can be shown that after placing job j, all gaps are of size at least p_j: By the induction hypothesis, after placing job $j-1$, all gaps were of size at least p_{j-1} which is at least p_j, by the assumed ordering of the jobs. Then, no matter how job j is placed, it is impossible that a gap of size smaller than p_j is created (see also Figure 3).

From now on, assume that n is even; the proof for the odd case is similar. Let A be the sum of the larger half of the sizes among $p'_1, \ldots, p'_n$ and B the sum of the smaller half. The truncation process reduces a job to the size of gap that follows it. Therefore, the makespan of the schedule produced by Greedy on p' (or on p) is $A + B$.

The previous claim implies that all sizes among $p'_1, \ldots, p'_n$ are within a ratio of two, which implies $A \leq 2B$.

From Lemma 1 we have $\mathrm{OPT}(p') \geq 2B$. Furthermore, we clearly have $\mathrm{OPT}(p') \leq \mathrm{OPT}(p)$. We can hence upper bound the makespan of the schedule produced by Greedy on p as

$$A + B \leq 3B \leq \frac{3}{2}\mathrm{OPT}(p') \leq \frac{3}{2}\mathrm{OPT}(p).$$

□

Note that this analysis did not use the fact that Greedy places jobs in the largest gap. The crucial property required in the analysis is the fact that when the placement of a job j increases the makespan, then all gaps are of size strictly less than $2p_j$.

We were not able to determine the exact approximation factor of Greedy. In Figure 5, an instance is illustrated that shows that the approximation factor of Greedy is at least 1.05: On the instance with jobs of processing times $20, 20, 10, 5, 5, 4, 4, 4, 4$, Greedy produces a schedule of makespan 42, by placing them in the order $20, 4, 10, 4, 20, 5, 5, 4, 4$, while the optimal schedule places the jobs in order $20, 5, 10, 5, 20, 4, 4, 4, 4$ and has makespan 40. This example gives the following lower bound.

Lemma 3. *The approximation ratio of Greedy is at least* 1.05*.*

We conducted a systematic search for stronger lower bound constructions, but could only obtain tiny improvements. For example, we found an instance consisting of 52 jobs showing a lower bound of $101/96 > 1.052$.

2.3 A case where Greedy is optimal

Theorem 1. *Greedy is optimal for instances with binary tree ratio at most* 2*.*

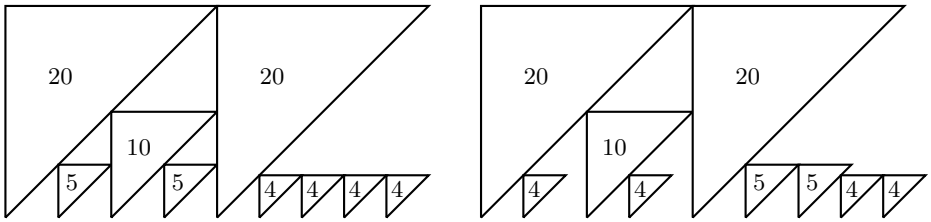

Fig. 5. The optimal schedule (left) and the schedule produced by Greedy (right) for the lower bound instance.

Proof. We show by induction on j that after placing job j, there are 2 gaps of size p_i, for every $\lceil j/2 \rceil + 1 \leq i < j$, and either a single gap (if j is odd) or 2 gaps (if j is even) of size $p_{(j+1)/2}$. This invariant is true after placing job 1, where there is a single gap of size p_1. When j is even, the job is placed in the single gap of size $p_{j/2}$. By the assumption on the binary tree ratio we have $2p_j \geq p_{j/2}$, implying that this gap is replaced by 2 gaps of size p_j. When j is odd, the job is placed in one of the 2 gaps of size $p_{(j-1)/2+1}$, and for the same reasons as in the even case the gap is replaced by 2 gaps of size p_j. In both cases the invariant is preserved.

The implication of this observation is that by the lower bound of Lemma 1, the schedule produced by Greedy is optimal. □

We can relate the jobs in a tree structure as illustrated in Figure 6. Job 1 is the root of the tree. Then for every job $j = 2, \ldots, n$, if Greedy inserted it into a gap assigned to job i, then job j is a descendant of job i. The result is a single root, connected to a binary tree, which is complete except possibly for the last level, which is left padded. The job labels on this tree are ordered by levels. The binary tree ratio of the instance is the maximum ratio between a job and its immediate ancestor in the tree, which was the motivation for the name of this ratio.

3 NP-hardness

Theorem 2. TS *is strongly NP-hard for instances with binary tree ratio strictly larger than* 2*.*

Proof. We reduce from the strongly NP-hard numerical 3-dimensional matching problem, see [7, problem SP16]. An instance of this problem consists of integers

$$a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n, D,$$

with $D \geq 4$, and for all i:

$$D/4 < a_i, b_i, c_i < D/2. \tag{1}$$

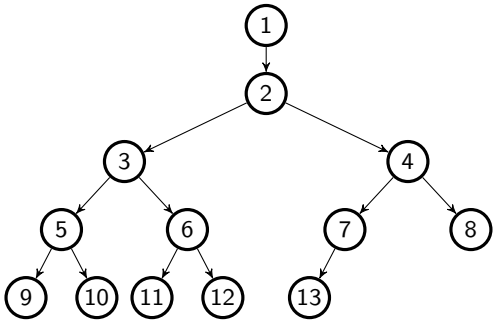

Fig. 6. Tree structure of the schedule produced by Greedy on any instance of 13 jobs with binary tree ratio at most 2.

Furthermore, we are guaranteed that

$$\sum_{i=1}^{n} a_i + b_i + c_i = nD. \tag{2}$$

The goal is to form n disjoint triplets of the form (i, j, k) with $a_i + b_j + c_k = D$.

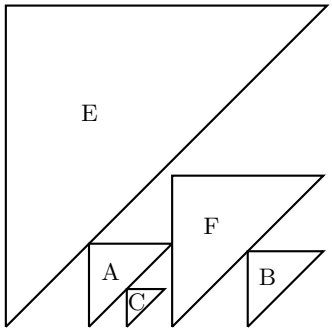

Fig. 7. A schematic view of a block in the optimal schedule obtained from the reduction.

Fix some arbitrary large constant $M \geq \frac{5D}{4}$. The instance consists of $5n$ jobs.

- There are n jobs E of size $8M + 5D$.
- There are n jobs F of size $4M$.
- For every $i \in \{1, \ldots, n\}$ there is a job A_i of size $2M + 2a_i + D$,
- as well as a job B_i of size $2M + b_i$,
- and a job C_i of size $M + c_i + D$.

We claim that the instance has a solution of makespan $n(8M + 5D)$ if and only if the initial numerical 2-dimensional matching instance has a solution.

For the easy direction, given a solution to the numerical 2-dimensional matching instance we construct a schedule consisting of the concatenation for every triplet (i, j, k) of the jobs E, A_i, C_k, F, B_j. Straightforward verification shows that the resulting schedule has the required makespan.

For the hard direction, consider a solution to TS of makespan $n(8M + 5D)$. Its makespan cannot be smaller, by the presence of the E jobs, that need to be scheduled every $8M + 5D$ time units. They structure the time line into n blocks of equal size $8M + 5D$ each.

Suppose that a block contains k jobs of size $x_1 \geq \ldots \geq x_k$ plus a single E job. These jobs create $k + 1$ gaps in this block. Each gap can be assigned to the smaller one among the neighboring jobs, and for an assignment we denote by $\alpha_i \in \{0, 1, 2\}$ the number of assignments the job of size x_i obtained.

Hence a lower bound on the block size is given by the expression $\alpha_1 x_1 + \ldots + \alpha_k x_k$ for some $\{0, 1, 2\}$-weights with $\alpha_1 + \ldots + \alpha_k = k + 1$. A valid lower bound is given by $\alpha_1 x_1 + \ldots + \alpha_k x_k$ setting α_i to 2 for the last $\lceil k/2 \rceil$ indices (corresponding to the smaller jobs), setting $\alpha_{k/2-1} = 1$ if k is even and setting $\alpha_i = 0$ to the remaining indices (see also Lemma 1). We will use this lower bound to determine the number of jobs of each type in a block.

No block can host two F jobs or more, since $3(4M) > 8M + 5D$. Hence every block contains exactly one F job.

No block can host an F and two A jobs, since (placing weights $\alpha_1 = \alpha_2 = 2$ for the A jobs and $\alpha_3 = 0$ for the F job)

$$4\left(2M + 2\frac{D}{4} + D\right) = 8M + 6D.$$

Hence every block contains exactly one F and one A job.

Similarly a block cannot not host an A and F job, together with two B jobs, as setting total α-weight 4 for the B jobs and 1 for the A job results in the lower bound

$$4\left(2M + \frac{D}{4}\right) + 2M + 2\frac{D}{4} + D = 10M + \frac{5}{2}D.$$

Hence every block contains exactly one F one A and one B job.

Finally a block cannot host an A, F and B job, together with two C jobs, as setting total α-weight 4 for the C jobs and 2 for the B job results in the lower bound

$$4\left(M + \frac{D}{4} + D\right) + 2\left(2M + 2\frac{D}{4}\right) = 8M + 6D.$$

In conclusion every block contains an F job, and A_i job, a B_j job and a C_k job with the following lower bound for the space occupied for these jobs, using α-weight 2 for jobs B_j, C_k and α-weight 1 for job A_i

$$2M + 2a_i + D + 2(2M + b_j) + 2(M + c_k + D) = 8M + 3D + 2(a_i + b_j + c_k).$$

Since this value cannot exceed the size of a block, namely $8M + 5D$, every block corresponds to a triplet (i, j, k) with $a_i + b_j + c_k \leq D$. By the assumption (2) we have equality for every triplet, and this shows that there is a solution to the numerical 2-dimensional matching instance, see Figure 7 for illustration.

By construction when the jobs are ordered in decreasing size, they are grouped by types, in the order F, E, A, B, C. Hence the binary tree ratio is determined by the ratio between an F and an E job, between a B and an F job, and between a C and an A jobs. All these fractions can be made arbitrarily close to 2 by choosing a large enough value for M. □

4 A QPTAS

Theorem 3. TS *admits a quasi polynomial time approximation scheme.*

Proof. The proof starts with a sequence of claims.

Claim. Rounding all sizes up to the nearest power of $1 + \epsilon$ changes the optimal makespan by at most a factor $1 + \epsilon$.

Proof. Take an optimal solution and multiply all processing times and start times by factor $1 + \epsilon$. The solution is still feasible, only the unit has changed. Now round all triangles down to the nearest power of $1 + \epsilon$ while keeping the start times fixed. □

Claim. We may assume that the ratio p_1/p_n is at most n/ϵ.

Proof. The optimal makespan OPT is at least p_1. All triangle with size less than $\epsilon p_1/n$ can be put at the end. We do not need to optimize over these, as they increase the makespan by at most an ϵ factor. □

From now let the smallest triangle have size $p_n = 1$ and the largest have size $p_1 \leq n/\epsilon$. Then $p_1 \leq \text{OPT} \leq np_1 \leq n^2/\epsilon$.

Claim. We may assume there are only $\lceil \log(n/\epsilon) \rceil + 1$ number of different sizes.

The proof follows from the previous two claims.

Claim. Restricting start times to values from the set $P = \{0, K, 2K, ..., \lceil n^2/\epsilon \rceil K\}$ for $K = \epsilon p_1/n$ increases the optimal makespan at most by factor $1 + \epsilon$.

Proof. We modify the start times of the jobs, processing them from left to right. For every job, we move it to the next time in P, and move simultaneously all subsequent jobs by the same amount to preserve feasibility. The value of the solution increases by at most $nK \leq \epsilon p_1 \leq \epsilon\text{OPT}$. □

Let Z be the set of all different job sizes after the above rounding. We have

$$|Z| = \lceil \log(n/\epsilon) \rceil + 1$$
$$|P| \leq \lceil n^2/\epsilon \rceil.$$

We design a dynamic programming scheme as follows. Partition the set of jobs into sets S and its complement $\overline{S}$. We want to compute the optimal schedule placing first jobs from $\overline{S}$ and then jobs from S. Only a few parameters from the first schedule influence the possibilities of the second schedule, namely the position of the rightmost triangle of each size. Hence we describe every possible configuration by a vector from P^Z, which leads to a quasi polynomial number of configurations. The number of possible subsets S is (in terms of size multiplicities) $O(n^Z)$ which is also quasi-polynomial.

Define $F(C, S)$ to be the optimal makespan of a schedule, placing first $\overline{S}$ with a configuration C and placing then jobs from S. The goal is to compute $F(e, \{1, \ldots, n\})$, where e is the empty configuration, which for technical reasons assigns to each size $x \in Z$ the starting time $-x$.

The basis cases consist of $F(C, \emptyset)$, where C ranges over all configurations from P^Z, some of them might be infeasible. Then $F(C, S)$ can be computed from $F(C + j, S - j)$ where $S - j$ is S minus one triangle $j \in S$ and $C + j$ is obtained from C by adding the triangle j to the right of C and placing it as early as possible, i.e. at time $\max_{x \in Z} C_x + x$. The number of choices for j (in terms of size) is at most $|Z|$. Hence, we need at most $|Z|$ look-ups to compute one value. □

5 Final remarks

We introduced a new scheduling problem, motivated by mixed criticality. The novelty lies in its combinatorial structure which defines the contribution to the makespan of each job in a non-local manner. We showed that the problem is strongly NP-hard, thus ruling out the existence of a fully polynomial time approximation scheme under the assumption P≠NP. In addition we provided a quasi polynomial time approximation scheme, ruling out APX-hardness, again under the assumption P≠NP. Furthermore, we introduced a greedy algorithm for this problem, but still do not understand well its approximation ratio. Closing the gap between the lower bound of 1.05 and the upper bound of 1.5 is the main question left open by this paper.

We would like to thank Marek Chrobak and Neil Olver for helpful discussions.

This work is partially supported by PHC VAN GOGH 2015 PROJET 33669TC, the grants FONDECYT 11140566, ANR-15-CE40-0015, and by the Project AI & Reasoning CZ.02.1.01/0.0/0.0/15_003/0000466 as well as by the European Regional Development Fund. Christian Konrad is supported by Icelandic Research Fund grants 120032011 and 152679-051.

References

1. Sanjoy Baruah, Vincenzo Bonifaci, Giuseppe D'Angelo, Haohan Li, Alberto Marchetti-Spaccamela, Nicole Megow, and Leen Stougie. Scheduling real-time mixed-criticality jobs. *IEEE Transactions on Computers*, 61(8):1140–1152, 2012.

2. Sanjoy Baruah, Haohan Li, and Leen Stougie. Towards the design of certifiable mixed-criticality systems. In *Proceedings of the 2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, RTAS '10, pages 13–22, Washington, DC, USA, 2010. IEEE Computer Society.
3. Cyril Briand, H Trung La, and Jacques Erschler. A robust approach for the single machine scheduling problem. *Journal of Scheduling*, 10(3):209–221, 2007.
4. Alan Burns and Rob Davis. Mixed criticality systems: A review. Technical report, 7th edition, University of York, 2015.
5. Amy Chou. NP-hard triangle packing problems. Research Science Institute summer program for highschool students, January 2016.
6. Jan Dvorak and Zdenek Hanzalek. Multi-variant time constrained flexray static segment scheduling. In *Factory Communication Systems (WFCS), 2014 10th IEEE Workshop on*, pages 1–8. IEEE, 2014.
7. Michael R Garey and David S Johnson. Computers and intractability: a guide to NP-completeness, 1979.
8. Chuancai Gu, N. Guan, Qingxu Deng, and Wang Yi. Improving OCBP-based scheduling for mixed-criticality sporadic task systems. In *2013 IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 247–256, Aug 2013.
9. Zdeněk Hanzálek and Tomáš Pácha. Use of the fieldbus systems in an academic setting. In *Proceedings of the Third IEEE Real-Time Systems Education Workshop*, RTEW '98, pages 93–, Washington, DC, USA, 1998. IEEE Computer Society.
10. Zdeněk Hanzálek, Tomáš Tunys, and Přemysl Šůcha. An analysis of the non-preemptive mixed-criticality match-up scheduling problem. *Journal of Scheduling*, 19(5):601–607, 2016.
11. SJ Honkomp, L Mockus, and GV Reklaitis. Robust scheduling with processing time uncertainty. *Computers & Chemical Engineering*, 21:S1055–S1060, 1997.
12. Taeju Park and Soontae Kim. Dynamic scheduling algorithm and its schedulability analysis for certifiable dual-criticality systems. In *Proceedings of the Ninth ACM International Conference on Embedded Software*, EMSOFT '11, pages 253–262, New York, NY, USA, 2011. ACM.
13. D. Socci, P. Poplavko, S. Bensalem, and M. Bozga. Mixed critical earliest deadline first. In *2013 25th Euromicro Conference on Real-Time Systems*, pages 93–102, July 2013.
14. Michael Stock. Canaerospace specification, 2014.
15. Steve Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, pages 239–243. IEEE, 2007.
16. M Wilson. *Robust scheduling in an uncertain environment*. PhD thesis, TU Delft, Delft University of Technology, 2016.